제22회 한국정보처리학회 추계학술발표대회 논문집 제 11권 제2호 (2004. 11)

품질 속성 기반의 소프트웨어 아키텍처 평가 프로세스

손이경*, 김행곤*, 현창문**
*대구가톨릭대학교 컴퓨터정보통신공학부
**탐라대학교 게임개발학과

e-mail: *{twilly°, hangkon}@cu.ac.kr
**cmhyun@cheju.tamna.ac.kr

Software Architecture Evaluation Process Based on Quality Attribute

Lee-Kyeong Son°*, Haeng-Kon Kim*, Chang-Moon Hyun**
*Dept. of Computer Information Communication,
Catholic University of Daegu
**Dept. of Game Development, Tamna University

요 약

소프트웨어 기술은 웹과 인터넷의 대중화로 소프트웨어의 품질 개선과 다양한 요구 변화에 효율적으로 대응하기 위해 급격히 발전하고 있는 추세이다. 완성된 시스템이 다양한 스테이크 홀더들의 품질에 대한 요구를 만족시키는 시스템인지의 여부를 결정하는 소프트웨어 아키텍처의 평가는 매우 중요한 부분이다. 소프트웨어 아키텍처는 프로젝트 초기의 설계 결정사항에 대한 산출물로써 시스템과 프로젝트에 많은 영향을 미치며 특정 시스템의 품질 속성은 주로 소프트웨어 아키텍처에 의해 결정된다. 본 논문에서는 품질 속성을 기반으로 체계적으로 아키텍처를 평가할 수 있는 프로세스를 제시하고 이 프로세스를 NextGen POS 시스템에 적용시켜 보았다.

1. 서론

소프트웨어 개발 기술은 사용자들의 요구사항 급변과 품질 요구의 증가에 따라 이에 효율적으로 대응할 수 있는 방법을 찾기 위해 급격히 발전하고 있으며 시스템의 규모와 복잡도도 현저히 증가함에 따라 시스템을 분할하고 구조화하여 시스템의 성능을 향상시킬 수 있는 소프트웨어 아키텍처 개념이 중요하게 여겨지고 있다. 이때 완성된 시스템이 다양한 스테이크 홀더들의 요구를 만족시키는 시스템인지의 여부를 결정하는 소프트웨어 아키텍처의 평가는 매우 중요한 부분이다. 시스템의 생명주기 초반에 소프트웨어 품질을 평가하면 비용 효율이 높아지게 되고, 품질 요구를 만족하지 못하는 대량의 자원이 시스템 구축에 사용될 수도 있다. 이러한 이유로, 시스템이 구축되기 전에 시스템 품질을 만족하는지의 여부를 평가하고 결정하는 것은 매우 중요하다[1].

본 논문에서는 소프트웨어의 품질 속성을 기반으로 소프트웨어 아키텍처를 평가하는 프로세스를 제시하고 NextGen POS 시스템에 적용시켜 보았다. 이 프로세스는 크게 요구 분석 단계, 아키텍처 뷰 표현 단계, 설계 결정 검색 및 분석 단계, 종합적 검증 단계로 구성되며, 각 단계마다 세부적인 작업을 수행함에 따라 소프트웨어 아키텍처의 평가에 중요한 영향을 미치는 품질 속성을 중심으로 한 체계적인 아키텍처 평가가 이루어질 수 있다[2].

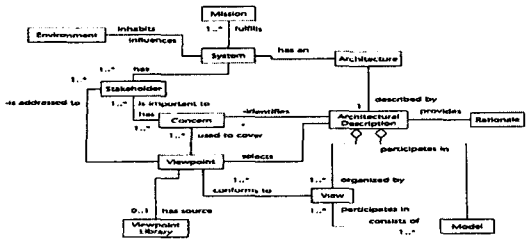
본 연구는 대구광역시 2004년 "이공계 우수공학 연구센터" 지원사업에 의해 지원됨.

2. 관련 연구

2.1 소프트웨어 아키텍처

소프트웨어 아키텍처란 소프트웨어 컴포넌트, 이들 컴포넌트들의 가시적인 속성, 그리고 컴포넌트들 사이의 관계로 구성된 시스템의 전체적인 구조로써 시스템을 설계하고 발전시키기 위한 지침과 원리라 정의할 수 있다.

소프트웨어 아키텍처는 다양한 스테이크 홀더들 간의 원활한 의사소통의 수단으로 시스템을 위한 기술적인 청사진의 역할과 프로젝트 초기 설계 결정사



(그림 1) 아키텍처 개념적 구성요소

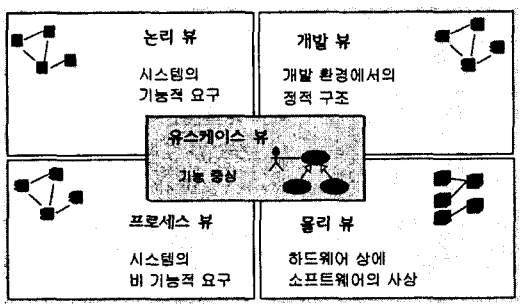
항을 기재해 놓은 산출물로써 시스템의 품질 속성과 개발 조직의 구성에 많은 영향을 미치며 여러 시스템을 위해 재사용할 수 있는 시스템의 추상적 모습으로서의 역할을 한다. IEEE 1471에 의한 소프트웨어 아키텍처의 개념적 구성요소는 (그림1)과 같다[3].

또한 소프트웨어 아키텍처는 내포하고 있는 구조가 너무나 복잡하여 1차원적으로 설명하기 곤란하므로 시스템의 여러 측면을 고려하기 위하여 다양한 관점을 바탕으로 정의된다[4]. Kruchten의 4+1 뷰 모델(그림 2)은 4개의 뷰로 각 스테이크 홀더에게 적절한 설계 결정을 획득하게 하고 다섯 번째 뷰를 통해 각각을 설명하고 확인한다[5].

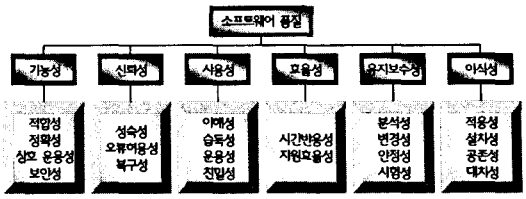
2.2 품질 속성(Quality Attribute)

소프트웨어 품질 속성이란 시스템이 만족해야하는 품질에 대한 요구사항을 의미한다. 품질 속성을 만족하지 못하는 자원이 시스템 구축에 사용됨으로써 시스템 전체가 품질에 대한 요구를 만족하지 못하는 경우가 발생하게 된다. 이러한 이유로 시스템 구축 전에 품질에 대한 요구를 만족하는지를 평가하고 결정하는 것은 매우 중요하며 이는 소프트웨어 아키텍처의 평가에도 많은 영향을 미친다.

품질 요소는 크게 시스템 품질 요소, 비즈니스 품질 요소, 아키텍처 품질 요소로 나누어지며 ISO 9126에서는 소프트웨어의 품질 요소를 (그림3)과 같이 6가지로 분류하고 있다.



(그림 2) 4+1 뷰 모델



(그림 3) 소프트웨어 품질 특성

3. 소프트웨어 아키텍처 평가 프로세스

3.1 개 요

소프트웨어 개발에 관련된 대부분의 활동들이 아키텍처에 의해 계획되기 때문에 가능한 빠른 시점에 아키텍처가 가지고 있는 문제점을 찾는 것이 좋다.

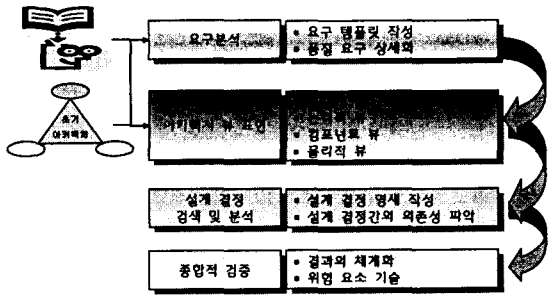
본 논문에서는 요구 템플릿과 품질 요구 명세서를 작성하는 요구 분석단계, 4+1 뷰 모델로 아키텍처를 표현하는 아키텍처 뷰 표현 단계, 뷰들로부터 품질 속성에 많은 영향을 미치는 설계결정을 찾고 분석하는 설계결정 검색 및 분석 단계, 마지막으로 지금까지의 결과물을 체계적으로 분류하고 잠재적 위험 요소를 식별하는 종합적 검증 단계로 구성되는 소프트웨어 아키텍처 평가 프로세스를 제시한다.(그림3)

이러한 과정을 거쳐 소프트웨어 아키텍처에 대한 품질 속성을 예측하거나 품질 요구를 충족시킬 수 있도록 도와주며 품질 속성의 획득에 많은 영향을 미치는 아키텍처 설계결정이 체계적으로 발견되어 표현되며 이를 근거로 아키텍처 위험이 기술된다.

3.2 요구 분석 단계

3.2.1 요구 사항 템플릿 작성

평가에 포함되는 사용자의 요구를 기능적 요구(FRs)와 품질 요구(QRs)로 분리하여 문서화 할 수



(그림 4) 아키텍처 평가 프로세스

<표 1> 요구 분석 템플릿

속성	신뢰성(QA1), 수행능력(QA2), 융통성(QA3)		
문 맥	신뢰성(QA1)	QR1	
	수행능력(QA2)	QR2	
	융통성(QA3)	QR3, QR4	
기능적 요구 (FRs)	P1: Process sale		1
	M1: 지불 인증(credit, debit, check) M2: 외부 회계 시스템과 자동화된 오프라인 관계 M3: 실시간 트랜잭션 M4: 커스터마이즈된 pluggable 비즈니스 규칙 생성/실행		P1
	O1: 모든 세일를 기록하고 영구히 저장 O2: 모든 작업에 사용자 인증을 요구		P1
품질 요구 (QRs)	QR1: 외부 서비스를 사용할 수 없을 경우 판매를 완료하기위해 지역적인 해결책으로 해결을 노력 할 것 QR2: 구매자는 판매 시점이 매우 빠르게 진행되기를 희망함 QR3: NextGen POS 시스템의 서로 다른 고객들은 판매세정서 비즈니스 규칙이 유일해야 하므로 비즈니스가 pluggable해야 함 QR4: 서로 다른 고객들은 자신들의 POS 시스템이 다양한 네트워 구성을 요구함 (thick & thin 클라이언트, two-tier & N-tier 물리적 계층)		
QRs와 FRs의 관계	QR1	M1, M2, M3	
	QR2	M1, M3	
	QR3	M1, M3, M4	
	QR4	M1, M3	

있는 템플릿을 정의하는 단계로 평가에 의한 기대치가 조율되는 단계이다<표1>. 이때 주요 기능들을 상대적 중요도에 따라 나열함으로써 기능적 요구들에 대한 평가의 범위를 결정하고, 품질 요구와 기능 요구간의 관계를 기술함으로써 품질 요구에 대한 평가 범위를 결정한다. 마지막으로 각각의 품질 요구와 관련되어 있는 기능적 요구를 식별한다.

3.2.2. 품질 속성 상세화

아키텍처 평가에서 품질 속성에 대한 명확하고 유용한 특성을 관리하는 것이 중요하므로 <표2>와 같이 품질 속성에 대한 특성을 잘 표현할 수 있는 템플릿을 정의하고 사용한다.

3.3 아키텍처 뷰 표현 단계

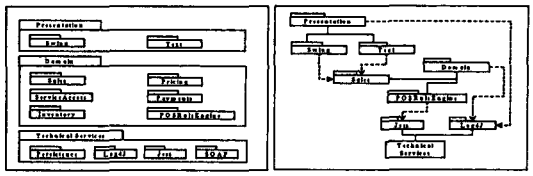
4+1 뷰 모델은 광범위한 애플리케이션에 대해 적절한 수준의 추상화를 가진 아키텍처를 기술하는데 도움이 되는 모델로써 각 뷰들마다 아키텍처 스타일과 패턴 표현이 가능하며 아키텍처와 레벨에 대한 정보의 일치로 더욱 명확한 평가가 가능하게 된다. (그림5)는 NextGen POS 시스템의 논리적인 계층 구조를 나타낸 것이며 (그림6)은 계층과 패키지간의 상호 작용을 나타낸 것이다.

<표 2> 품질 속성 특성표의 예

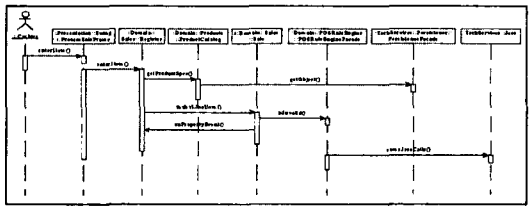
속성	요소	자극	설계 결정	반응
신뢰성	복구성	● 원격 서비스 실패 ● 원격 DB 실패	3.4.1에서 결정	오류 복구

속성	요소	자극	설계 결정	반응
수행능력	x	● 외부 지불 인증 시스템에서의 병목 현상 ● 빠른 판매 프로세싱	3.4.1에서 결정	보이지 않음

속성	요소	자극	설계 결정	반응
융통성	적용성	● Pluggable 비즈니스 룰 ● 제 3자 서비스	3.4.1에서 결정	다양성 보호
	구성능력	● 다양한 네트워크 구성 ● 구성의 수정	3.4.1에서 결정	변경에 따른 파급효과



(그림5) NextGen POS 시스템 논리구조



(그림6) 계층간 상호작용

3.4 설계 결정 검색 및 분석 단계

3.4.1 설계 결정 명세표 작성

아키텍처 설계 결정은 아키텍처 설계 동안 직면하게 될 설계상의 문제에 대한 해결책의 집합으로 이제까지 정의된 기능적 요구와 아키텍처를 바탕으로 유즈케이스를 식별하고 기능적 요구에 관련된 품질 속성들로부터 뷰를 결정하고 해당 뷰로부터 유즈케이스에 관련된 아키텍처 요소를 식별한다. 이와 함께 <표3>과 같이 결정 변수, 결정 값, 대안, 원리를 포함하는 아키텍처 설계 결정 명세표를 작성한다.

아키텍처 설계 결정은 품질 속성 획득에 많은 영향을 미치므로 아키텍처를 평가하는 동안 이를 검색하고 분석하는 작업은 매우 중요하다.

이렇게 얻어진 설계 결정들은 독립적으로 분석하여야 하며 분석 결과에 따라 특정한 품질 속성이 설계 결정에 미치는 효과와 원리에 대해 기술한다. 또한 <표2>의 설계 결정 부분도 완성시킬 수 있다.

<표 3> 아키텍처 설계 결정의 예

Decision:ADD1	결정 변수	결정 값	대안
Inter-Layer connection	LayersConnection	Using Facade	● Using Controller ● Using Observer
원리	● 원격 호출에 의해 야기되는 병목현상을 방지 ● 서브시스템들간의 의존성을 최소화함으로써 다른 플랫폼에 시스템을 간단하게 포팅할 수 있는 facade를 가정		
관련 아키텍처	See Fig 6		

Decision:ADD2	결정 변수	결정 값	대안
Large-scale logical structure	OverallStructure	Using layered structure	● Using pipeline ● Using blackboard
원리	● 추상화 수준의 증가를 기준으로 설계 ● 향상이나 변경을 지원 ● 호환성 지원		
관련 아키텍처	See Fig 5		

<표 4> 아키텍처 설계 결정들 간의 의존성

설계 결정		대체					
		ADD1	ADD2	ADD3	ADD4	ADD5	ADD6
주체	ADD1		(QA2,+) (QA3,+)	(QA3,+)	(QA2,+)	(QA2,+)	(QA2,+) (QA3,+)
	ADD2	(QA2,-) (QA3,+)		(QA3,+)	(QA2,-)	(QA2,-)	(QA2,-) (QA3,+)
	ADD3	(QA3,+)	(QA3,+)				(QA3,+)
	ADD4	(QA2,+)	(QA2,+)			(QA1,+) (QA2,+)	(QA1,+) (QA2,+)
	ADD5	(QA2,+)	(QA2,+)		(QA1,+) (QA2,+)		(QA1,+) (QA2,+)
	ADD6	(QA2,+) (QA3,+)	(QA2,+) (QA3,+)	(QA3,+)	(QA1,-) (QA2,+)	(QA1,-) (QA2,+)	

3.4.2 아키텍처 설계 결정간의 의존성 파악

각각의 설계 결정들은 다른 결정들에게 어떤 영향을 미치는가에 따라 식별될 수 있으며 이 단계를 통해 결정들 간의 의존성이 <표4>와 같이 나타난다. 이는 소프트웨어 아키텍트가 품질 속성들 간의 trade-off를 결정하고 아키텍처 설계를 변경함으로써 야기될 수 있는 효과를 이해하도록 도와준다. 이때 QAi는 특정 품질 속성을 뜻하며 +는 QA에서의 긍정적 효과를, -는 부정적 효과를 의미한다.

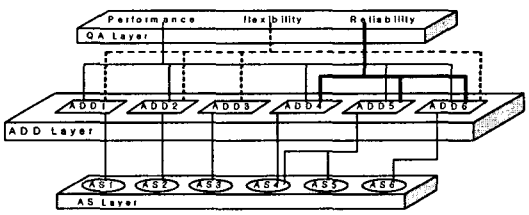
3.5 종합적 검증 단계

3.5.1 결과의 체계화

아키텍처와 품질 속성들 간의 관계를 체계적으로 분류하는 것은 아키텍처에서 품질 속성을 이해하고 제어하는 능력을 상당히 향상시킨다. 그러므로 (그림5)에서 보여주는 것 같이 3가지 종류의 계층(QA Layer, ADD Layer, AS Layer)으로 나누고 아키텍처 설계 결정(ADD Layer)을 통해 유용한 요소들을 종합적으로 조직화한다.

3.5.2 아키텍처 위험 요소 기술

마지막으로 잠재적으로 문제가 될 수 있는 아키텍처 설계 결정을 <표5>와 같이 원인과 결과, 아키텍처 위험 요소로 표현한다. 아키텍처 위험을 문서로 표현하는 것은 소프트웨어 아키텍트가 위험 요소를 인식하여 위험을 완화시킬 수 있는 계획을 세우도록 지원하기 위함이다.



(그림 7) 품질 요구와 아키텍처의 연결구조

<표 5> 아키텍처 위험 명세

RISK : RSK1	실행시간 효율성	ADD#	ADD2
관련된 속성	수행 능력		
원인	많은 계층을 통해 하나의 요구를 전송하는데 오버헤드 발생		
결과	실행 시간 효율성의 부족		
관련 아키텍처	Fig 6		

5. 결론 및 향후 연구

본 논문에서는 품질 속성을 만족시키는 소프트웨어의 개발을 위한 아키텍처 평가에 대한 과정을 제시하였다. 아키텍처의 평가는 소프트웨어에 대한 요구와 초기 아키텍처에 의해 시작되어 평가 과정을 수행하는 동안 요구 분석 템플릿, 품질 속성 명세표, 아키텍처 설계 결정, 아키텍처 의존 관계표 등과 같은 중간 결과물들을 정의 또는 생성하였으며 이를 근거로 아키텍처와 품질요구를 연결하는 종합 구조와 아키텍처 위험 요소들이 최종적으로 제공하였다.

그러므로 기존의 방법론에 비해 평가 과정이 잘 정의되어 있으며 UML의 뷰 모델을 사용하였고 평가 결과가 구체적으로 문서화 되었으므로 개념적인 흐름을 이해하는데 도움을 줄 뿐 아니라 아키텍처에 대한 명확한 통찰력을 얻음으로써 위험을 이해할 수 있게 한다. 향후에는 소프트웨어 요구에서의 모호성 처리와 좋지 않은 아키텍처 설계 결정의 처리와 같은 이슈들에 대해 관심 있게 논의할 것이다.

참고문헌

[1] Roger S. Pressman "Software Engineering A Practltiners' Approach" 3rd Ed. McGraw Hill

[1] P. Clements, R. Kazman, M, Klein, Evaluating Software Architecture: Methods and Case Studies, Addision-Wesley, 2002

[2] Bosch, J., Design and Use of Software Architecture, Addison-Wesley, 2000

[3] IEEE Std 1471-2000 Recommended Practice for Architecture Description of Software Intensive Systems, October 2000

[4] Mugurel T. Ionita, Dieter K. Hammer, Henk Obbink, Scenario-Based Software Architecture Evaluation Methods: An Overview, http://www.win.tue.nl/oas/architecting/aimes/papers

[5] Kruchten, P., "The 4+1 View Model of Software Architecture", IEEE Software, Vol. 12, No.6 pp.42-50. November 1995